Business Process Customization using Process Merging Techniques

Bulanov, Pavel; Lazovik, Alexander; Aiello, Marco

Published in:
International Conference on Service-Oriented Computing and Applications

Business Process Customization using Process Merging Techniques

Pavel Bulanov, Alexander Lazovik, Marco Aiello
Distributed Systems Group
University of Groningen
The Netherlands
Email: {p.bulanov, a.lazovik, m.aiello}@rug.nl

***Abstract*—One of the important application of service composition techniques lies in the field of business process management. Essentially a business process can be considered as a composition of services, which is usually prepared by domain experts, and many tasks still have to be performed manually. These include the design and creation of the process itself or the modification of an existing one when business requirements change. Any form of automation and support we can bring to the tasks of maintenance and evolution are highly beneficial.**

One way of creating a new business process is by the combination of two existing ones which naturally should retain the behavioral features of both original processes. In this paper, we introduce a formal language to express behavioral properties of processes together with its semantics, and we show how it supports process merging.

***Keywords*-Business process; Process evolution; Process maintenance; Process merging; Temporal logic**

I. Introduction

Business processes as well as service compositions capture with their definition the way in which an organization, possibly virtual, behaves and produces added value. The preciser the description of the process, the better can the organization be managed and operated. The process becomes part of the organization's knowledge in a number of ways. For instance, they can emerge naturally from practice, they can be designed by experts and refined after use or they can be mined based on some execution data. Furthermore, processes are not static abstractions, but rather they evolve together with the organization and with the environment in which they operate.

Consider the case of a special business process used and customized by more than one (sub)-organization and usually referred to as a **template**. Templates are usually made in order to accumulate the basic rules and recommendations of experts involved, which are applicable to most cases and therefore need to be preserved. Then, one of such templates can be used as is or it can be **customized** in order to address the specificities of a particular case (which may be driven by particular rules of a department of an affiliate, for example).

This is illustrated in Figure 1, with a template process in the upper left corner, and customized one in the upper right corner. The arrow "1" represents the customization of the template process. In such situation there is a potential danger of simultaneous modifications, when a template is modified after it has been customized (for example, in order to fix an error in the specification). Such modification is represented by arrow "2" in the figure. The problematic case in this situation is that the customized processes (there may be many of them) must be changed in order to take into account the template modification. The converging arrows "3" illustrate the fact that the new customized process is actually a derivative of both template and old customized processes.

Such combination of customized processes or service compositions can be automatized, but to do so, customization must be represented as a transformation function. One of the ways to do this, is by applying the technique which we propose here, namely, merging of business process models. Such merging can be made automatically and therefore can be repeated again once the template process is modified.

An example of such process evolution can be taken from the field of e–Government. There are 418 municipalities in the Netherlands, and they possess a certain amount of freedom in the application of laws, therefore, the software should be adopted to the local specifics as well [1], [2]. In the worst case there are 418 different customizations of the same business process, and all of them must be changes in case of the modifications in the central regulations.

Our study falls in the general area of having effective control of process modifications which is considered crucial for business process management (BPM), e.g. [3]. But most of the studies focus mainly on the problem of process flexibility and ease of changes in general, regardless of the nature of the process modifications [4], [5]. Though, the importance of merging of business processes was also pointed out in different studies [6], [7].

The main contribution of this paper is twofold. First, we introduce an idea to support business process evolution by application of process merging techniques. Second, in order to perform effective process merging, we go beyond well-known temporal logic formalisms for business process modeling, considering explicitly the existence of different kinds of branching points in the process structures. Also, the representation of a process in the terms of temporal formalisms gives the ability to make a combination of two processes on the basis on their temporal behaviors, which is more simple from the algorithmic point of view than to

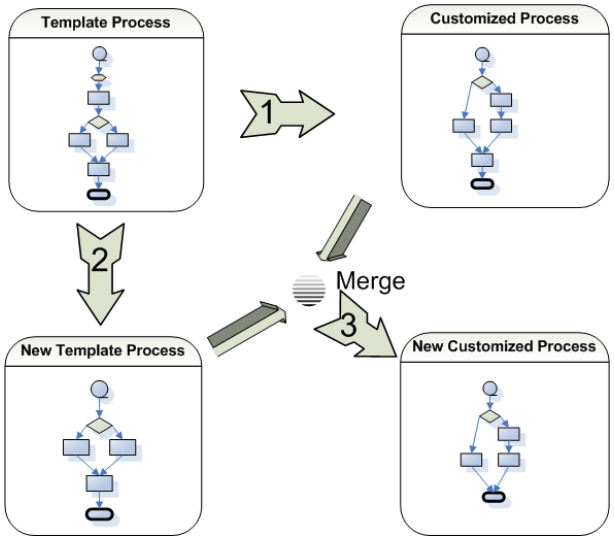

Figure 1. Process customization by merging

merge two processes represented, for instance, as graphs.

The rest of the paper is organized as follows. Section II provides the overview of the proposed approach. Formal definitions are represented in Section III. The formal description of process merging based on the proposed approach is discussed in Section IV. Related work is briefly addressed in Section V, and Section VI concludes the paper and outlines the future work.

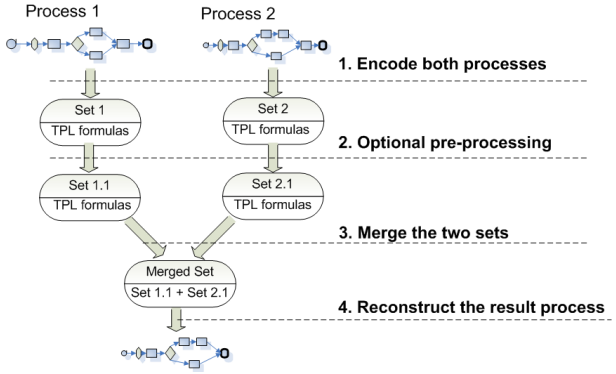

Figure 2. Merging process overview

II. Automatizing Process Merging

Starting from two existing business processes by merging them one can obtain a rich description including the mixed behaviors of both original ones. More precisely, the result process will retain the execution sequences of both parent processes.

To automatize the process of merging, we propose to encode the behaviors of the processes in a temporal logic like formalism and then perform the merging at the language level and to go then back by generating a new process from the formal description which is a consistent merge of the initial processes.

The overview of the automated merging is shown in Figure 2. The first step is to encode both processes in the terms of temporal process logic (TPL) formulas. Then, there is an optional pre–processing step, its purpose is to cure possible contradictions between the two processes basing on some predefined strategy, e.g., the processes may have different importance and the structural features of the less important one can be discarded. The next two steps are the core of the whole idea.

The third step is to merge the two sets of formulas, and the main benefit there is that the merging is merely the combination of two sets of formulas and can be made automatically. There are some restrictions though, which will be discussed later.

The fourth step is to re–construct the final process basing on the unified set of formulas, and later in this article we provide the algorithm of process reconstruction basing on a set of TPL formulas.

III. A Formal Language for Business Process Merging

Formal languages have often been the foundations for BPM automation. Much researched examples include Petri-Nets, Process calculi and Temporal Logic. Our approach uses the latter as the basis for defining a language prone to process merging automation. We begin by defining the formal language for processes, then we provide a semantics for the language and move onto giving algorithms for the merging procedure.

A. Temporal Process Logic

The Temporal Process Logic (TPL) is a modal propositional language that talks about the truth of propositions in future states, but also about possible execution runs. The underlying processes are considered to have AND and OR gates in addition to simple branching from one state to another one. Its syntax is quite straightforward: we have a set of proposition symbols AP, propositional unary and binary operators $\neg, \wedge, \vee$, plus three unary modal operators $\rightarrow, \rightsquigarrow$. The intuitive meaning of the last operators is the following.

- $\rightarrow a$ means there is a state satisfying a in the process and that this state can always be reached from the current state following the process model. In case of parallel splitting, at least one of the parallel branches must lead to the state satisfying a.
- $\rightsquigarrow a$ means there is a state satisfying a in the process, and this state can (but not always) be reached from the current state following the process model. The non–determinism appears there due to the nature of OR–gates, when it is not possible to tell in advance the actual execution path in the run time.

The syntax of TPL is therefore defined as follows:

1) Each propositional logic formula is a formula of TPL;
2) If ϕ is a formula of TPL, so is $\neg\phi$;
3) If ϕ and ψ are formulas of TPL, so are $\phi \wedge \psi$ and $\phi \vee \psi$;
4) If ϕ is a formula of TPL, so are $\rightarrow \phi$ and $\rightsquigarrow \phi$.

B. Process Runs as TPL Semantics

The idea is to evaluate TPL formulas over processes and to use them to describe the behavior of the processes. We begin by providing a formal definition of a process with AND and OR gates. This definition actually reflects a formalization of a business process specified in terms of BPMN–like notation.

Definition 1 (Process): A process P is a tuple $\langle A, G, T\rangle$ where:

- *A is a finite set of activities, with selected start activity $\odot$ and final activity $\otimes$. Each activity, apart from the start and final ones, represents a single call of a service.*
- *G is a finite set of gateways, each of type AND or OR;*
- *$S = A \cup G$ is a set of steps;*
- *$T = T_a \cup T_g$, where:*
- *$T_a : (A \backslash \{\otimes\}) \rightarrow S$ is a finite set of transitions, which assign a next step for each activity;*
- *$T_g : G \rightarrow 2^S$ is a finite set of transitions, which assign a nonempty set of next steps for each gateway.*

Later when talking about a process P we will refer to its set of activities as P_A, its set of gateways as P_G, and its set of transitions as P_T. The next entity we need is a process run in the process P.

Definition 2: A process run σ of a process P is a sub–graph of the graph P, built in the following way:

1) *All activities and gateways of P remain in σ;*
2) *For any activity, its transition to the next step remains in σ;*
3) *For any AND–gateway, all its transition to the next steps remain in σ;*
4) *For any OR–gateway, one or more of its transition to the next steps (transitions are picked non–deterministically) remain in σ;*

In other words, a process run represents a single execution of an abstract business process engine such that for each OR–gate a different path is taken while for each AND–gate all subsequent sub–paths are included.The set of all possible process runs of a process P is denoted as Ω_P or just Ω.

We say that an activity a is followed by an activity b w.r.t. the process run σ and denote that as $a <_\sigma b$ when there is a path in the graph σ leading from a to b. We say that an activity a is included into a process run σ and denote is as $a \in \sigma$ if $\odot <_\sigma a$.

C. TPL Truth definition

We can now establish the link between TPL formulas and the process models. Let AP be a set of propositional variables, and a labeling function $\nu : P_A \rightarrow 2^{AP}$ which assigns each variable in AP with a set of activities where that variable holds true. A model $\mathcal{M}$ is a pair $\langle P, \nu\rangle$, where P is a process and ν is a valuation function. Now we can define the truth of a TPL formula in a model, note that the truth is local to activities of the process.

$$
\begin{array}{lcl}
\mathcal{M}, x \models a & \Leftrightarrow & x \in \nu(a) \\
\mathcal{M}, x \models \neg a & \Leftrightarrow & \mathcal{M}, x \text{ not } \models a \\
\mathcal{M}, x \models a \vee b & \Leftrightarrow & \mathcal{M}, x \models a \text{ or } \mathcal{M}, x \models b \\
\mathcal{M}, x \models \rightarrow b & \Leftrightarrow & \forall \sigma \in \Omega_P : x \in \sigma \Rightarrow \\
 & & \exists y \in \sigma : x <_\sigma y \wedge \mathcal{M}, y \models b \\
\mathcal{M}, x \models \rightsquigarrow b & \Leftrightarrow & \exists \sigma \in \Omega_P : x \in \sigma \Rightarrow \\
 & & \exists y \in \sigma : x <_\sigma y \wedge \mathcal{M}, y \models b
\end{array}
$$

It this paper we also use some syntactical enhancements, namely, we use the shorthand form $a \rightsquigarrow b$ instead of $a \Rightarrow \rightsquigarrow b$, and $a \rightarrow b$ instead of $a \Rightarrow \rightarrow b$.

D. Process Construction

Given a set of TPL formulas one could consider all the propositional variables appearing in them and build a model where each one of these is true in exactly one state. Such a model in general would not be minimal, but would be a clear representation of a business process that can be then executed. Let us call $\mathring{a}$ a proposition letter that is true exactly at activity a and $\widehat{P} = \langle P, L\rangle$ a model built only of these activities.

The task of simplified model checking can be defined as follows: build a model $\widehat{P}$, such as all formulas are valid in all states of that model, or, $\forall \phi \in S, \forall a \in P : \widehat{P}, a \models \phi$. Such kind of model checking is simplified because the class of models is limited to the one described in the previous paragraph.

Consider a set of TPL formulas, all of which are of kind $a \circ b$, where a and b are propositional letters, and $\circ$ is either $\rightsquigarrow$ or $\rightarrow$. For a set of such formulas it is possible to build a process due to resemblance with a typical graph representation. Only two issues must be taken into account:

(1) There may be redundant information in the set of formulas, which need to be eliminated with the help of transitive reduction of the original set.

(2) There are two types of formulas, $\rightarrow$ and $\rightsquigarrow$, but either of them is represented as a link between nodes. Therefore, gateways (AND– or OR– ones) must be introduced.

IV. Process Merging

Previously in Section II we have already given the informal introduction into the process merging technique. Now we are in the position to formalize it and provide an algorithm for process merging.

Briefly the merging technique consists of the following steps:

1) Represent both processes in the form of TPL formulas;
2) (Optional) pre–process both sets of TPL formulas in order to reveal and cure possible contradictions;

3) Merge the sets and get their union;
4) Construct the final process basing on that union.

The first step was not discussed yet, but since the model construction can only operate the formulas of type $a \circ b$, then it is natural to represent a process in such kind of formulas only. More precisely, for each two activities A and B we check if either of $A \rightarrow B$ or $A \rightsquigarrow B$ is valid, and if yes, then include such formula in the set. If both of those formulas are valid, then include the stronger one (that is, $A \rightarrow B$).

The second step involves the identification of the formulas which are contradicting each other. Once identified, such formulas should be modified or removed in a way which preserves the process structure. This can be done by hand, or semi-automated. The third step is merely a union of two sets, and the fourth step is a process reconstruction according to the algorithm outlined in the previous section.

V. Related Work

La Rosa et al. [6] propose the idea of merging of business process models basing on the similarities between regions in business process. In other words, they outline the similar parts in the BPs being merged and work out a new process model which conforms to both original process models.

Also, Sun et al. [8] offer another approach to BP merging which is based on WF–Nets [9] . The approach is based on the similarity between WF–Nets, which is in turn based on the number of the atomic changes needed in order to convert a process A into a process B.

Gottschalk et al. [10] offer a merging technique where for each business process a corresponding graph is built, and the merging is reduced to the merging of the graphs. The approach is general, though the technique offered in our paper has simpler representation structure (linked sets instead of graphs).

In contrast with the studies mentioned above, our idea relies on the temporal relations thus allowing to reveal the similarities which are not based on geometrical similarity but rather on the similarity in the behavior. Also, representation of a process as a set of temporal relations allows to make a combination between a traditional BPMN–like process with a declarative one [4].

This gives the following advantages: i) compatibility with variability techniques (e.g., [1]), such as declarative–based specification of a template process; ii) it allows to mix declarative and imperative process specifications in one system; iii) it is open for refinements through the pre–processing step.

The idea to utilize the temporal relations in order to deal with BP management was studied in [11] and [12], but these studies are focused mainly on the flexibility and change management in business processes.

VI. Concluding Remarks

Formal methods to represent processes, provide strong advantages when managing the evolution of the process themselves. This includes also the addition of behaviors and possibly the merging with other processes. We have presented a formal language to describe process behaviors and its underlying process models. Among the advantages of the proposal is the algorithmically simple merging procedure as well as the ability to extend the patterns for merging via several pre–processing steps.

Acknowledgements

The research is supported by the NWO **SaS-LeG** project, http://www.sas-leg.net, contract No. 638.001.207.

References

[1] M. Aiello, P. Bulanov, and H. Groefsema, "Requirements and tools for variability management," in *IEEE workshop on Requirement Engineering for Services (REFS 2010) at IEEE COMPSAC*, 2010.

[2] N. R. T. P. van Beest, P. Bulanov, H. Wortmann, and A. Lazovik, "Resolving business process interference via dynamic reconfiguration," in *ICSOC*, 2010, pp. 47–60.

[3] W. M. P. van der Aalst and S. Jablonski, "Dealing with workflow change: identification of issues and solutions," *International Journal of Computer Systems Science and Engineering*, vol. 15, no. 5, pp. 267–276, September 2000.

[4] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W. M. P. van der Aalst, "Process flexibility: A survey of contemporary approaches," in *CIAO! / EOMAS*, ser. LNBIP, vol. 10. Springer, 2008, pp. 16–30.

[5] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features - enhancing flexibility in process-aware information systems," *Data Knowl. Eng.*, vol. 66, no. 3, pp. 438–466, 2008.

[6] M. L. Rosa, M. Dumas, R. Uba, and R. M. Dijkman, "Merging business process models," in *OTM Conferences (1)*, 2010, pp. 96–113.

[7] J. M. Küster, C. Gerth, A. Förster, and G. Engels, "A tool for process merging in business-driven development," in *CAiSE Forum*, 2008, pp. 89–92.

[8] S. Sun, A. Kumar, and J. Yen, "Merging workflows: A new perspective on connecting business processes," *Decision Support Systems*, vol. 42, no. 2, pp. 844–858, 2006.

[9] W. M. P. van der Aalst, "The application of petri nets to workflow management," *Journal of Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[10] F. Gottschalk, W. M. P. van der Aalst, and M. H. Jansen-Vullers, "Merging event-driven process chains," in *OTM Conferences (1)*, 2008, pp. 418–426.

[11] R. Lu, S. Sadiq, and G. Governatori, "On managing business processes variants," *Data Knowl. Eng.*, vol. 68, no. 7, pp. 642–664, 2009.

[12] M. Pesic, M. H. Schonenberg, N. Sidorova, and W. M. P. van der Aalst, "Constraint-based workflow models: Change made easy," in *OTM Conferences (1)*, 2007, pp. 77–94.